Figure_1

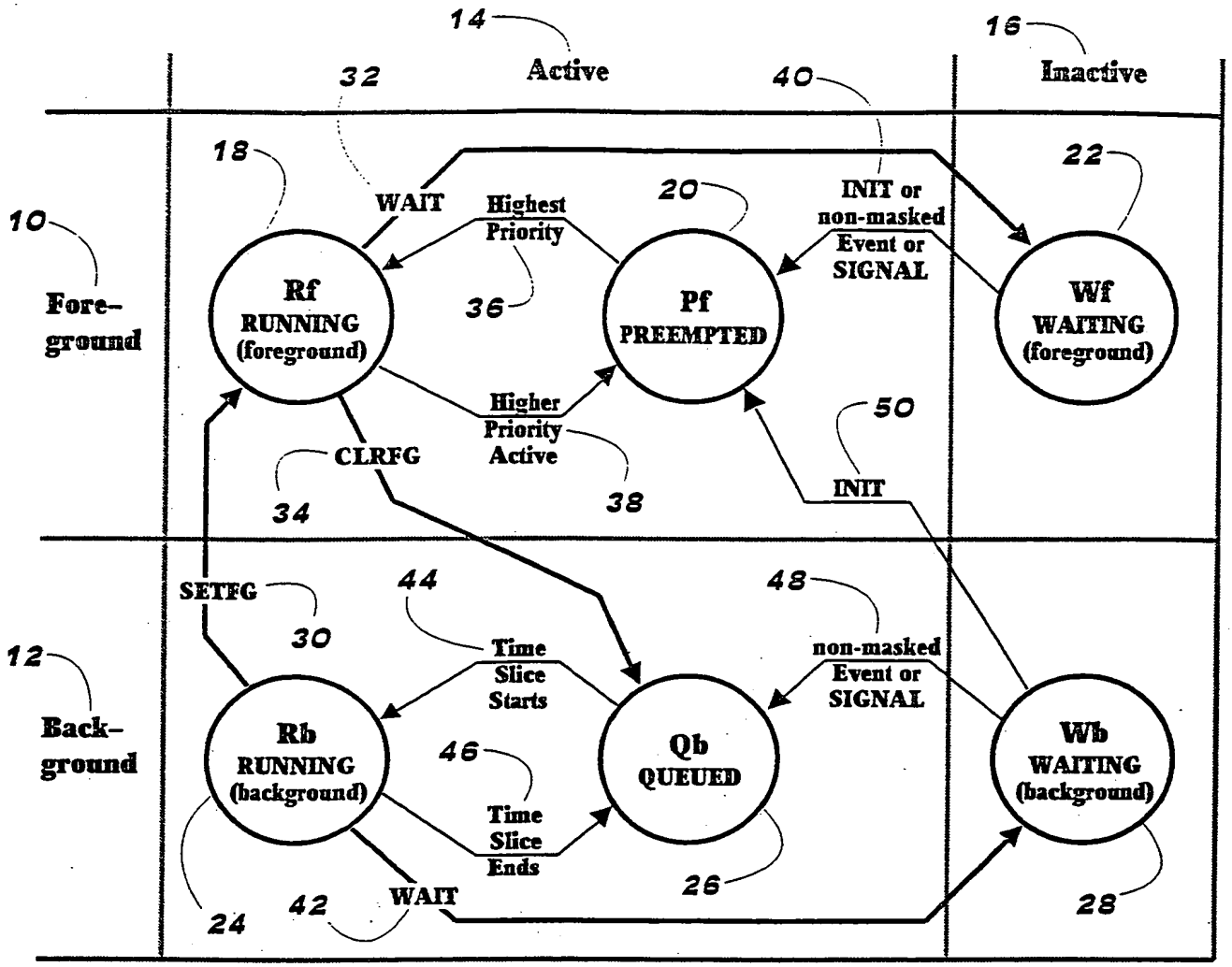

14
Active
16
Inactive
32
40
18
22
10
WAIT
Highest
Priority
20
INIT or
non-masked
Event or
SIGNAL
Fore-
ground
Rf
RUNNING
(foreground)
Pf
PREEMPTED
Wf
WAITING
(foreground)
36
Higher
Priority
Active
50
CLRFG
34
38
INIT
SETFG
30
44
48
12
Time
Slice
Starts
non-masked
Event or
SIGNAL
Back-
ground
Rb
RUNNING
(background)
Qb
QUEUED
Wb
WAITING
(background)
46
Time
Slice
Ends
26
24
42
WAIT
28

Figure_2

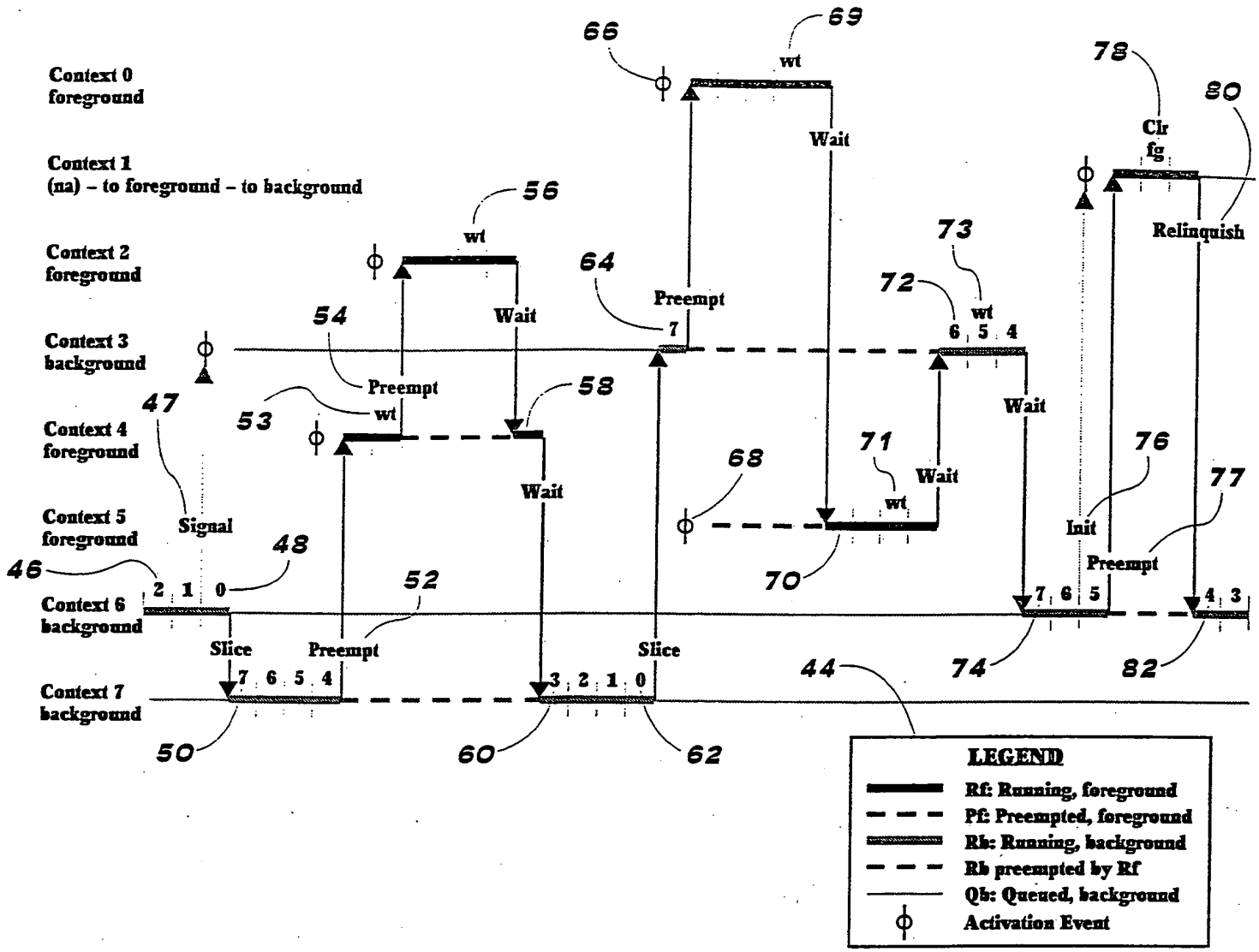

Context 0 foreground
Context 1 (na) – to foreground – to background
Context 2 foreground
Context 3 background
Context 4 foreground
Context 5 foreground
Context 6 background
Context 7 background
66
69
wt
Wait
78
80
Clr fg
Relinquish
56
wt
64
Preempt
7
73
72
wt
6 5 4
54
Wait
Preempt
47
53
wt
58
Wait
Signal
71
wt
Wait
68
76
77
Init
Preempt
46
2 1 0
48
52
70
7 6 5
4 3
Slice
Preempt
Slice
7 6 5 4
3 2 1 0
74
82
44
50
60
62
LEGEND
Rf: Running, foreground
Pf: Preempted, foreground
Rb: Running, background
Rb preempted by Rf
Qb: Queued, background
Activation Event

Figure_3

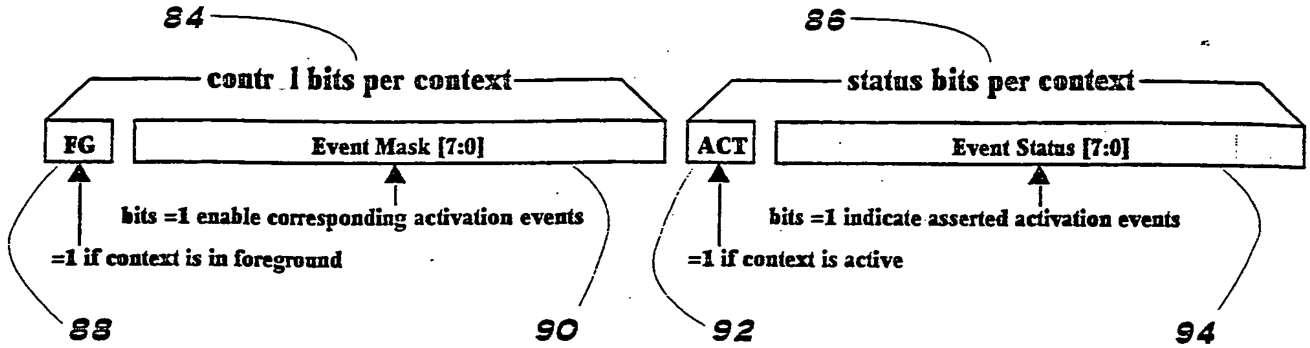

84
contr_l bits per context
86
status bits per context
FG
Event Mask [7:0]
ACT
Event Status [7:0]
bits =1 enable corresponding activation events
=1 if context is in foreground
bits =1 indicate asserted activation events
=1 if context is active
88
90
92
94

Figure_4

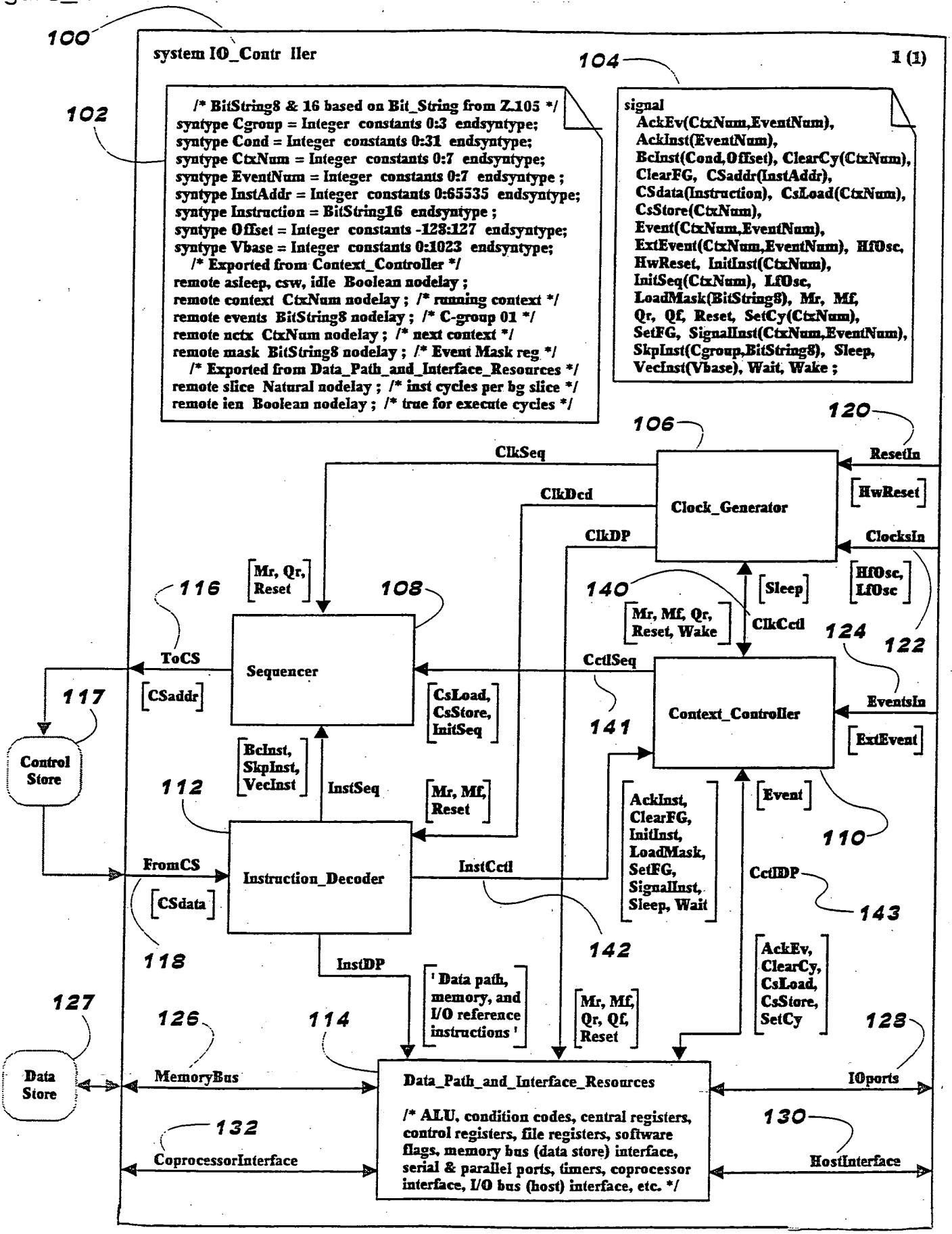

100
system IO_Contr ller
1 (1)
102
/* BitString8 & 16 based on Bit_String from Z.105 */
syntype Cgroup = Integer constants 0:3 endsyntype;
syntype Cond = Integer constants 0:31 endsyntype;
syntype CtxNum = Integer constants 0:7 endsyntype;
syntype EventNum = Integer constants 0:7 endsyntype ;
syntype InstAddr = Integer constants 0:65535 endsyntype;
syntype Instruction = BitString16 endsyntype ;
syntype Offset = Integer constants -128:127 endsyntype;
syntype Vbase = Integer constants 0:1023 endsyntype;
/* Exported from Context_Controller */
remote asleep, csw, idle Boolean nodelay ;
remote context CtxNum nodelay ; /* running context */
remote events BitString8 nodelay ; /* C-group 01 */
remote nctx CtxNum nodelay ; /* next context */
remote mask BitString8 nodelay ; /* Event Mask reg */
/* Exported from Data_Path_and_Interface_Resources */
remote slice Natural nodelay ; /* inst cycles per bg slice */
remote ien Boolean nodelay ; /* true for execute cycles */
104
signal
AckEv(CtxNum,EventNum),
AckInst(EventNum),
BcInst(Cond,Offset), ClearCy(CtxNum),
ClearFG, CSaddr(InstAddr),
CSdata(Instruction), CsLoad(CtxNum),
CsStore(CtxNum),
Event(CtxNum,EventNum),
ExtEvent(CtxNum,EventNum), HfOsc,
HwReset, InitInst(CtxNum),
InitSeq(CtxNum), LfOsc,
LoadMask(BitString8), Mr, Mf,
Qr, Qf, Reset, SetCy(CtxNum),
SetFG, SignalInst(CtxNum,EventNum),
SkpInst(Cgroup,BitString8), Sleep,
VecInst(Vbase), Wait, Wake ;
106
Clock_Generator
ClkSeq
ClkDcd
ClkDP
120
ResetIn
HwReset
ClocksIn
HfOsc,
LfOsc
Sleep
ClkCctl
140
Mr, Mf, Qr,
Reset, Wake
122
124
Mr, Qr,
Reset
116
108
Sequencer
ToCS
CSaddr
CsLoad,
CsStore,
InitSeq
CctlSeq
141
Context_Controller
EventsIn
ExtEvent
117
Control
Store
BcInst,
SkpInst,
VecInst
InstSeq
112
Mr, Mf,
Reset
AckInst,
ClearFG,
InitInst,
LoadMask,
SetFG,
SignalInst,
Sleep, Wait
Event
110
FromCS
Instruction_Decoder
InstCctl
CctlDP
143
CSdata
118
142
InstDP
' Data path,
memory, and
I/O reference
instructions '
Mr, Mf,
Qr, Qf,
Reset
AckEv,
ClearCy,
CsLoad,
CsStore,
SetCy
127
126
114
128
Data
Store
MemoryBus
Data_Path_and_Interface_Resources
/* ALU, condition codes, central registers,
control registers, file registers, software
flags, memory bus (data store) interface,
serial & parallel ports, timers, coprocessor
interface, I/O bus (host) interface, etc. */
IOports
130
132
CoprocessorInterface
HostInterface

Figure_5

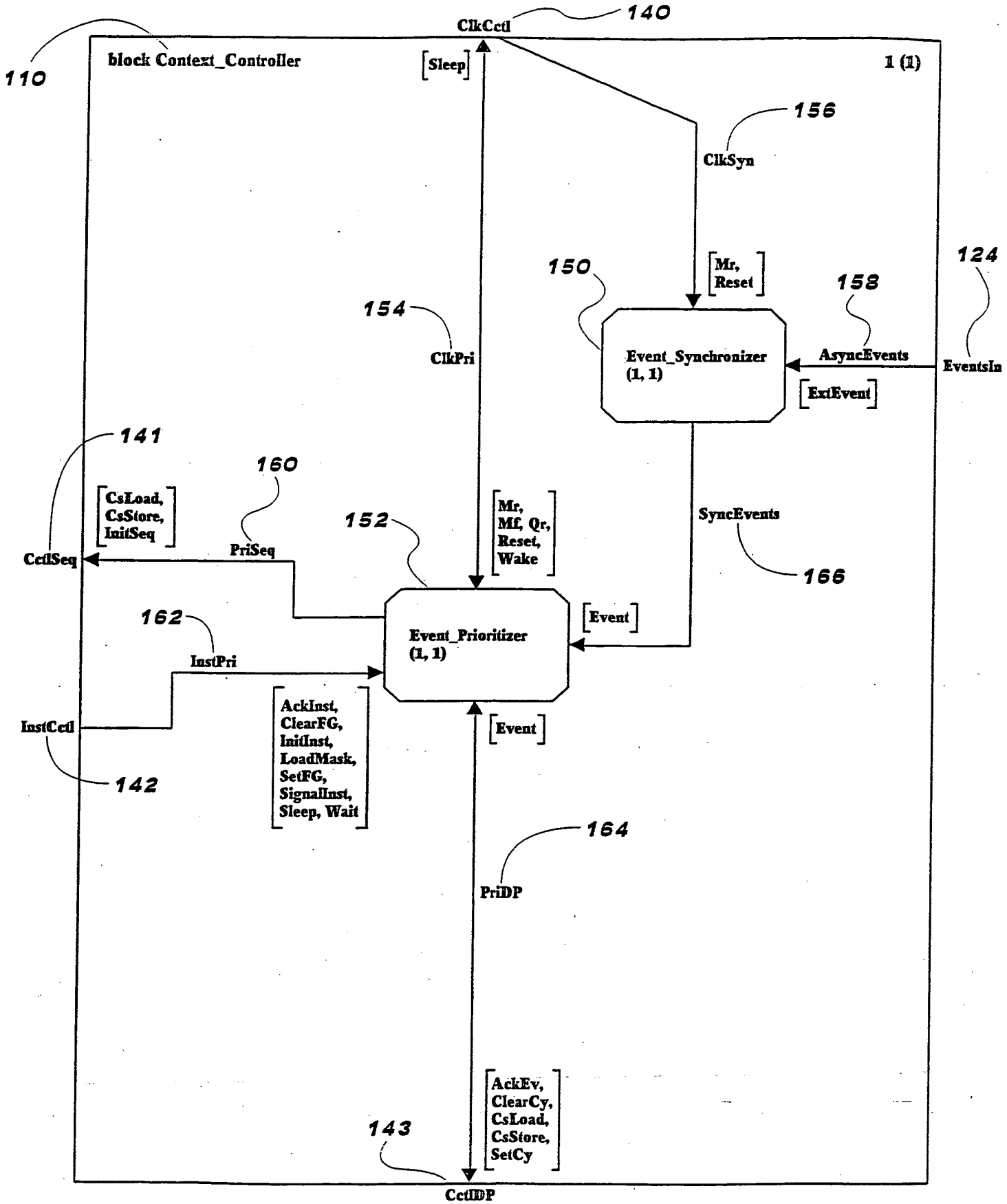

ClkCctl
140
110
block Context_Controller
[Sleep]
1 (1)
156
ClkSyn
150
[Mr, Reset]
124
158
154
ClkPri
Event_Synchronizer (1, 1)
AsyncEvents
EventsIn
[ExtEvent]
141
160
[CsLoad, CsStore, InitSeq]
152
[Mr, Mf, Qr, Reset, Wake]
SyncEvents
CctlSeq
PriSeq
166
162
[Event]
Event_Prioritizer (1, 1)
InstPri
[AckInst, ClearFG, InitInst, LoadMask, SetFG, SignalInst, Sleep, Wait]
InstCctl
[Event]
142
164
PriDP
[AckEv, ClearCy, CsLoad, CsStore, SetCy]
143
CctlDP

Figure_6

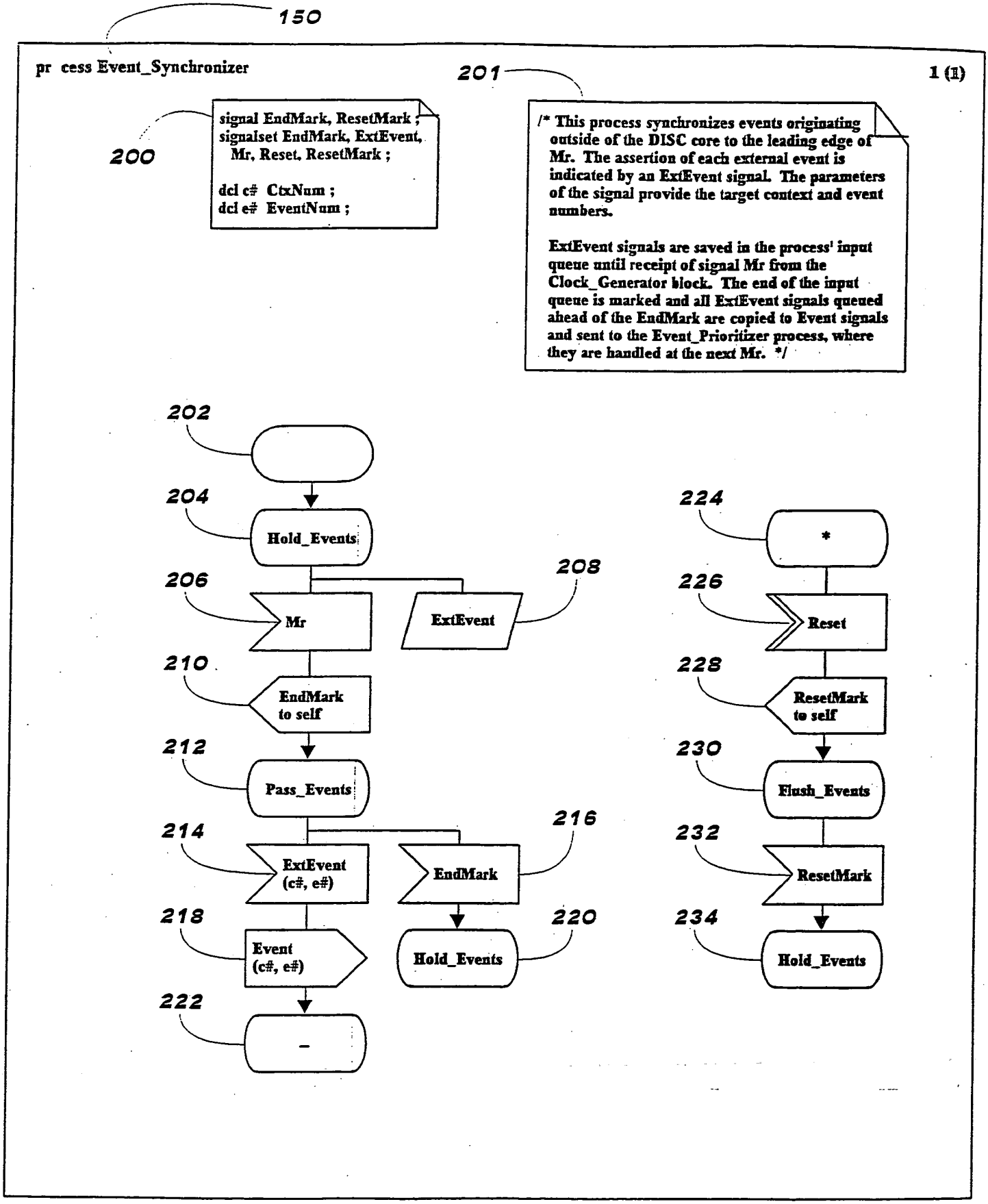

150
pr cess Event_Synchronizer
1 (1)
201
200
signal EndMark, ResetMark ;
signalset EndMark, ExtEvent,
Mr, Reset, ResetMark ;
dcl c# CtxNum ;
dcl e# EventNum ;
/* This process synchronizes events originating outside of the DISC core to the leading edge of Mr. The assertion of each external event is indicated by an ExtEvent signal. The parameters of the signal provide the target context and event numbers.
ExtEvent signals are saved in the process' input queue until receipt of signal Mr from the Clock_Generator block. The end of the input queue is marked and all ExtEvent signals queued ahead of the EndMark are copied to Event signals and sent to the Event_Prioritizer process, where they are handled at the next Mr. */
202
204
Hold_Events
206
Mr
208
ExtEvent
210
EndMark to self
212
Pass_Events
214
ExtEvent (c#, e#)
216
EndMark
218
Event (c#, e#)
220
Hold_Events
222
224
*
226
Reset
228
ResetMark to self
230
Flush_Events
232
ResetMark
234
Hold_Events

Figure_7A

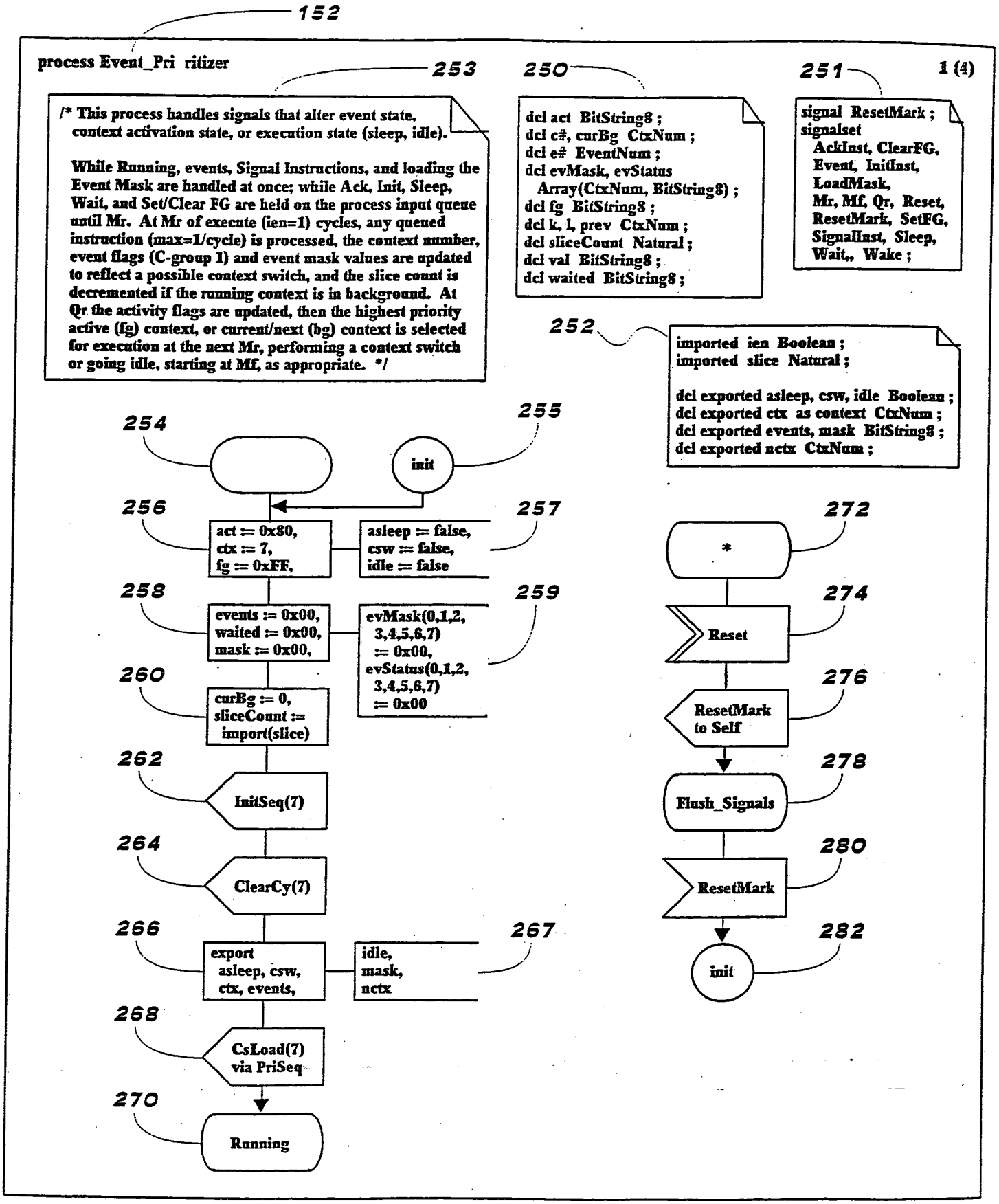

152
process Event_Pri ritizer
253
/* This process handles signals that alter event state, context activation state, or execution state (sleep, idle).
While Running, events, Signal Instructions, and loading the Event Mask are handled at once; while Ack, Init, Sleep, Wait, and Set/Clear FG are held on the process input queue until Mr. At Mr of execute (ien=1) cycles, any queued instruction (max=1/cycle) is processed, the context number, event flags (C-group 1) and event mask values are updated to reflect a possible context switch, and the slice count is decremented if the running context is in background. At Qr the activity flags are updated, then the highest priority active (fg) context, or current/next (bg) context is selected for execution at the next Mr, performing a context switch or going idle, starting at Mf, as appropriate. */
250
dcl act BitString8 ;
dcl c#, curBg CtxNum ;
dcl e# EventNum ;
dcl evMask, evStatus
Array(CtxNum, BitString8) ;
dcl fg BitString8 ;
dcl k, l, prev CtxNum ;
dcl sliceCount Natural ;
dcl val BitString8 ;
dcl waited BitString8 ;
251
signal ResetMark ;
signalset
AckInst, ClearFG,
Event, InitInst,
LoadMask,
Mr, Mf, Qr, Reset,
ResetMark, SetFG,
SignalInst, Sleep,
Wait,, Wake ;
1 (4)
252
imported ien Boolean ;
imported slice Natural ;
dcl exported asleep, csw, idle Boolean ;
dcl exported ctx as context CtxNum ;
dcl exported events, mask BitString8 ;
dcl exported nctx CtxNum ;
254
255
init
256
act := 0x80,
ctx := 7,
fg := 0xFF,
257
asleep := false,
csw := false,
idle := false
258
events := 0x00,
waited := 0x00,
mask := 0x00,
259
evMask(0,1,2,
3,4,5,6,7)
:= 0x00,
evStatus(0,1,2,
3,4,5,6,7)
:= 0x00
260
curBg := 0,
sliceCount :=
import(slice)
262
InitSeq(7)
264
ClearCy(7)
266
export
asleep, csw,
ctx, events,
267
idle,
mask,
nctx
268
CsLoad(7)
via PriSeq
270
Running
272
*
274
Reset
276
ResetMark
to Self
278
Flush_Signals
280
ResetMark
282
init

Figure_7B

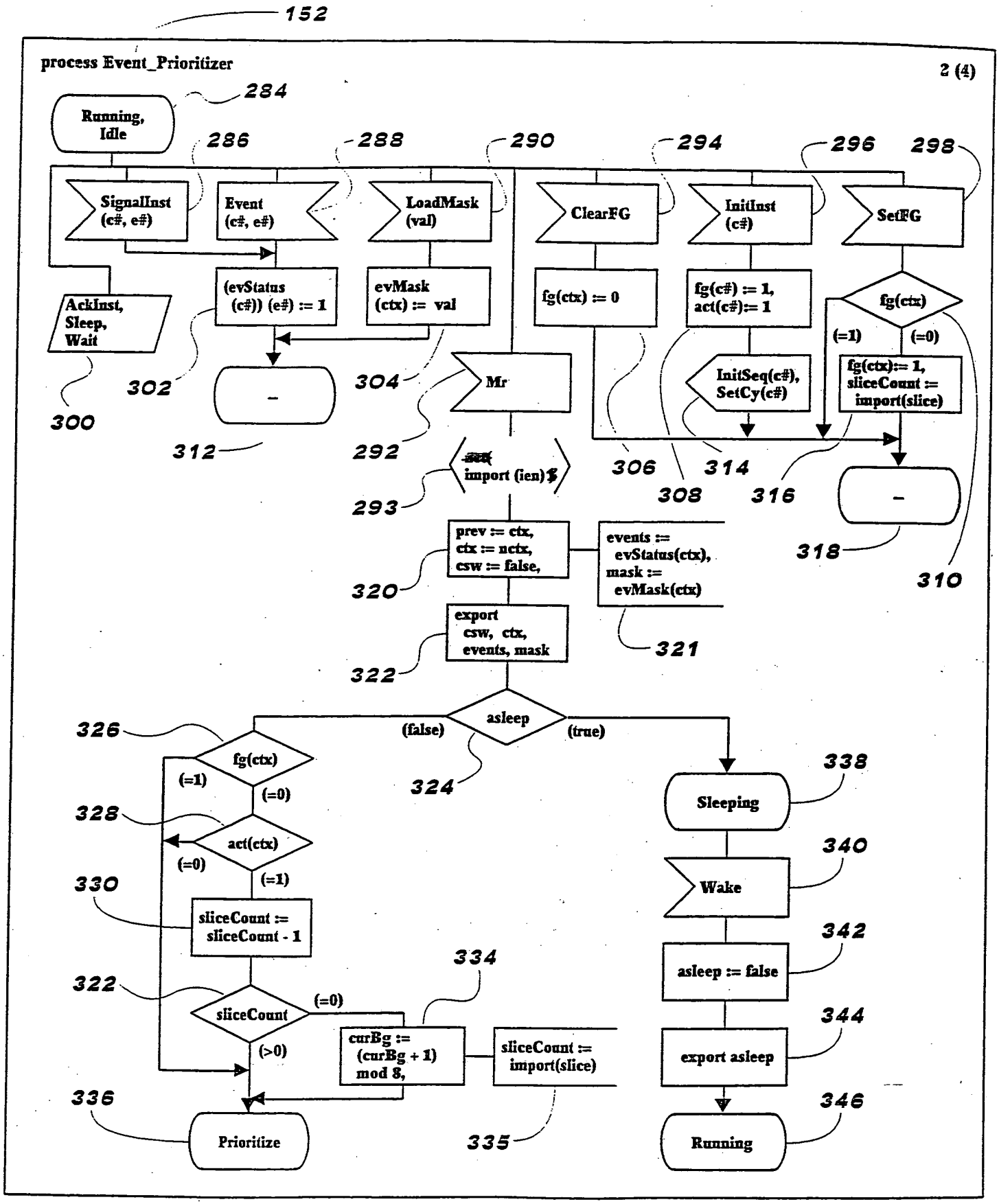

152
process Event_Prioritizer
2 (4)
284
Running, Idle
286
288
290
294
296
298
SignalInst (c#, e#)
Event (c#, e#)
LoadMask (val)
ClearFG
InitInst (c#)
SetFG
AckInst, Sleep, Wait
(evStatus (c#)) (e#) := 1
evMask (ctx) := val
fg(ctx) := 0
fg(c#) := 1, act(c#):= 1
fg(ctx)
(=1)
(=0)
fg(ctx):= 1, sliceCount := import(slice)
InitSeq(c#), SetCy(c#)
300
302
304
306
308
310
312
314
316
318
Mr
import (ien)
292
293
prev := ctx, ctx := nctx, csw := false,
events := evStatus(ctx), mask := evMask(ctx)
320
321
export csw, ctx, events, mask
322
asleep
(false)
(true)
324
326
fg(ctx)
(=1)
(=0)
328
act(ctx)
(=0)
(=1)
330
sliceCount := sliceCount - 1
322
sliceCount
(=0)
(>0)
334
curBg := (curBg + 1) mod 8,
sliceCount := import(slice)
335
336
Prioritize
338
Sleeping
340
Wake
342
asleep := false
344
export asleep
346
Running

Figure_7C

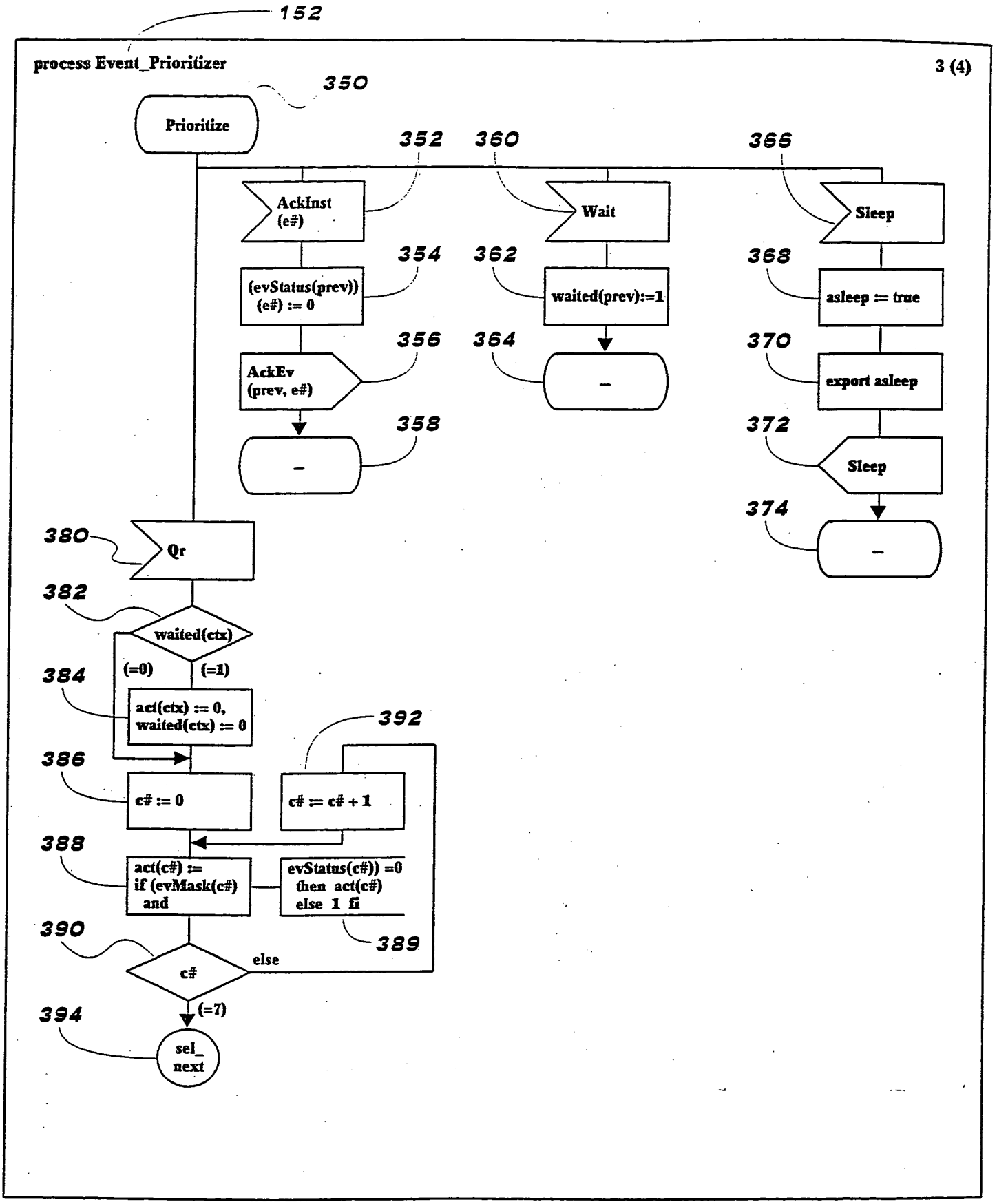

152
process Event_Prioritizer
3 (4)
350
Prioritize
352
360
365
AckInst
(e#)
Wait
Sleep
354
362
368
(evStatus(prev))
(e#) := 0
waited(prev):=1
asleep := true
356
364
370
AckEv
(prev, e#)
–
export asleep
358
372
–
Sleep
374
–
380
Qr
382
waited(ctx)
(=0)
(=1)
384
act(ctx) := 0,
waited(ctx) := 0
392
386
c# := 0
c# := c# + 1
388
act(c#) :=
if (evMask(c#)
and
evStatus(c#)) =0
then act(c#)
else 1 fi
389
390
c#
else
(=7)
394
sel_
next

Figure_7D

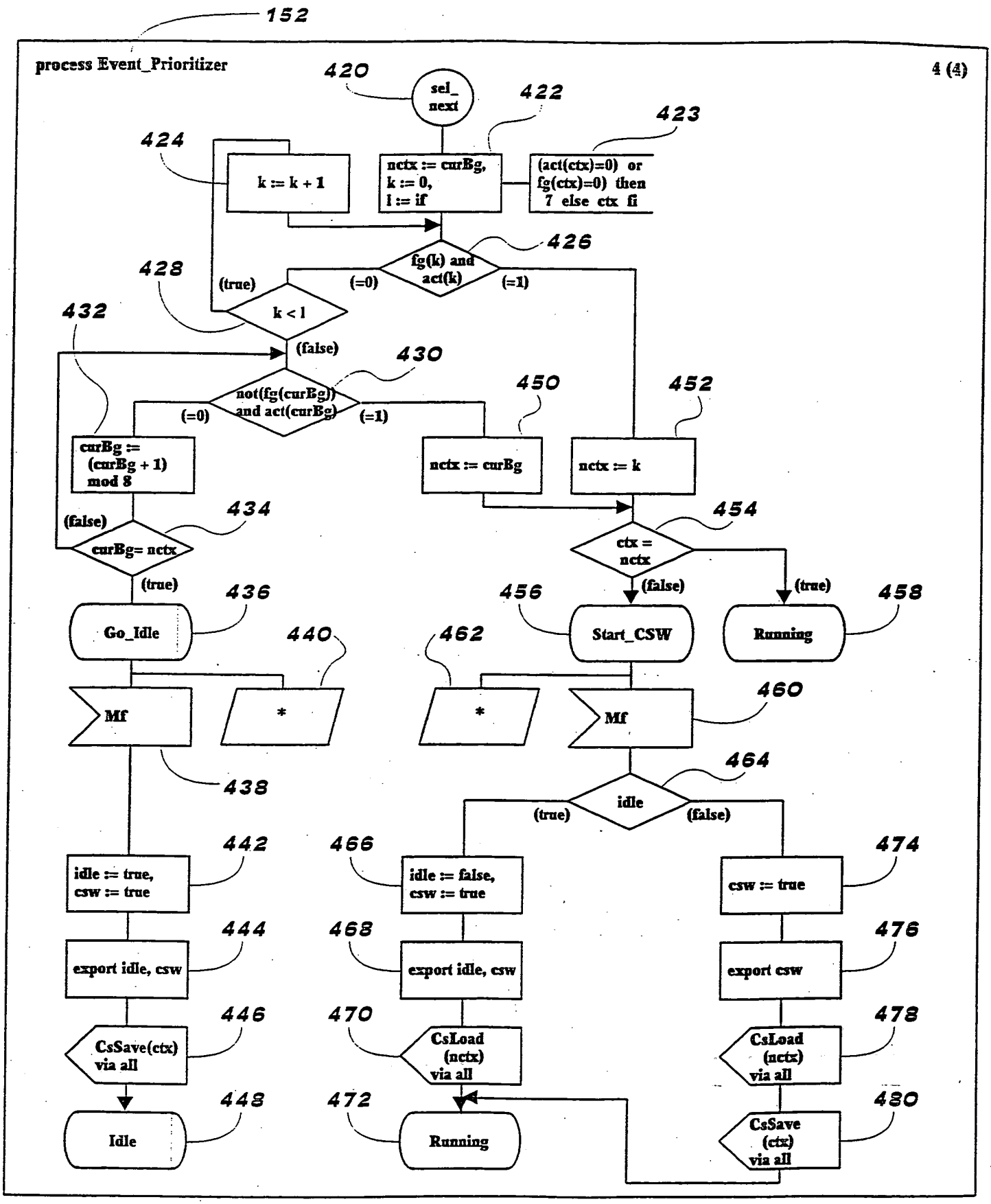

152
process Event_Prioritizer
4 (4)
420
sel_ next
422
423
424
k := k + 1
nctx := curBg, k := 0, l := if
(act(ctx)=0) or fg(ctx)=0) then 7 else ctx fi
426
fg(k) and act(k)
(=0)
(=1)
(true)
428
k < l
(false)
432
430
not(fg(curBg)) and act(curBg)
(=0)
(=1)
450
452
curBg := (curBg + 1) mod 8
nctx := curBg
nctx := k
(false)
434
curBg= nctx
(true)
454
ctx = nctx
(false)
(true)
436
456
458
Go_Idle
Start_CSW
Running
440
462
460
Mf
*
*
Mf
438
464
idle
(true)
(false)
442
466
474
idle := true, csw := true
idle := false, csw := true
csw := true
444
468
476
export idle, csw
export idle, csw
export csw
446
470
478
CsLoad(ctx) via all
CsLoad (nctx) via all
CsLoad (nctx) via all
448
472
480
Idle
Running
CsSave (ctx) via all

Figure_8

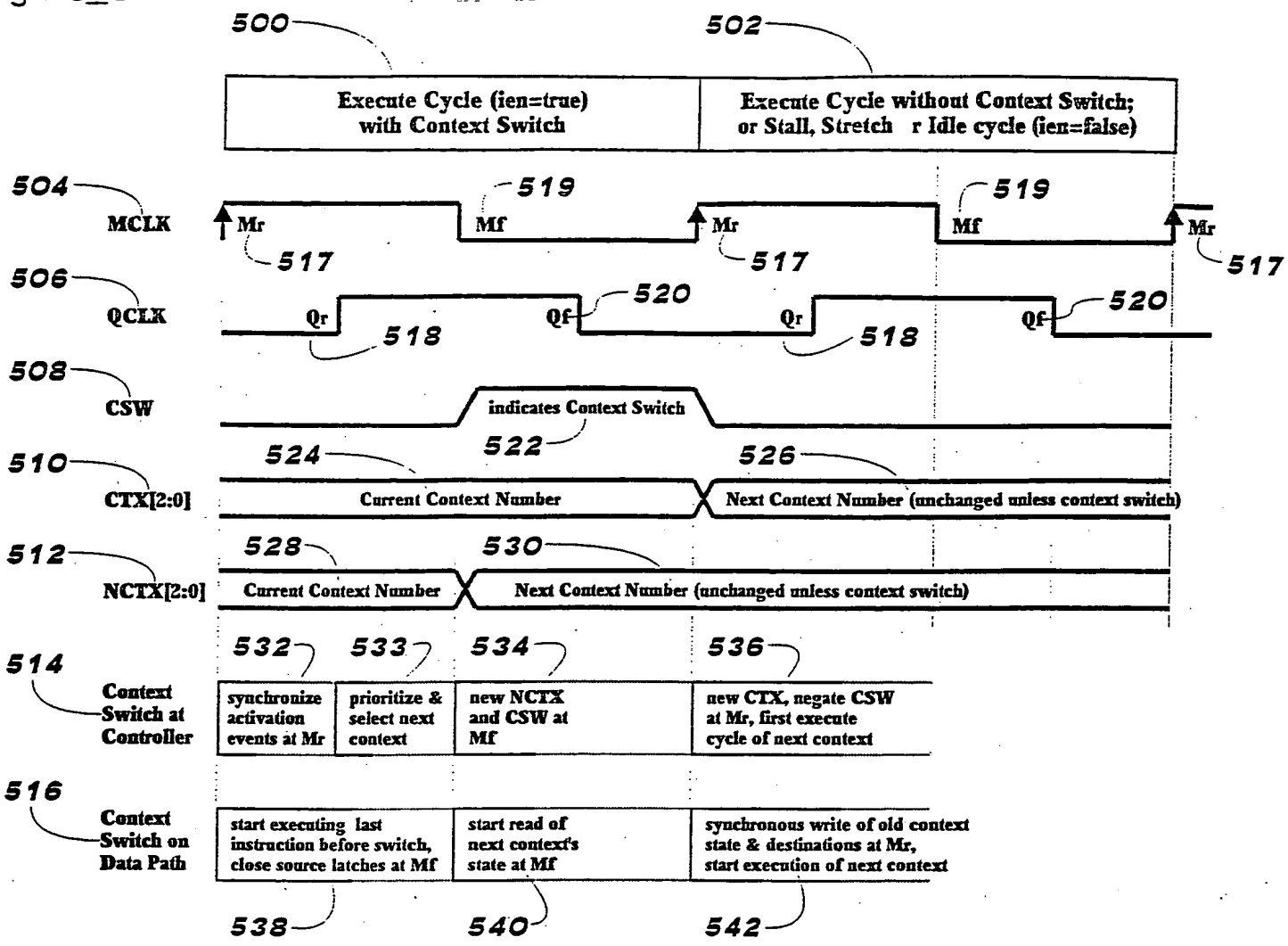

500
502
Execute Cycle (ien=true) with Context Switch
Execute Cycle without Context Switch; or Stall, Stretch r Idle cycle (ien=false)
504
MCLK
Mr
519
Mf
517
506
QCLK
Qr
518
Qf
520
508
CSW
indicates Context Switch
522
510
CTX[2:0]
524
Current Context Number
526
Next Context Number (unchanged unless context switch)
512
NCTX[2:0]
528
530
514
Context Switch at Controller
532
synchronize activation events at Mr
533
prioritize & select next context
534
new NCTX and CSW at Mf
536
new CTX, negate CSW at Mr, first execute cycle of next context
516
Context Switch on Data Path
start executing last instruction before switch, close source latches at Mf
start read of next context's state at Mf
synchronous write of old context state & destinations at Mr, start execution of next context
538
540
542

Figure_9

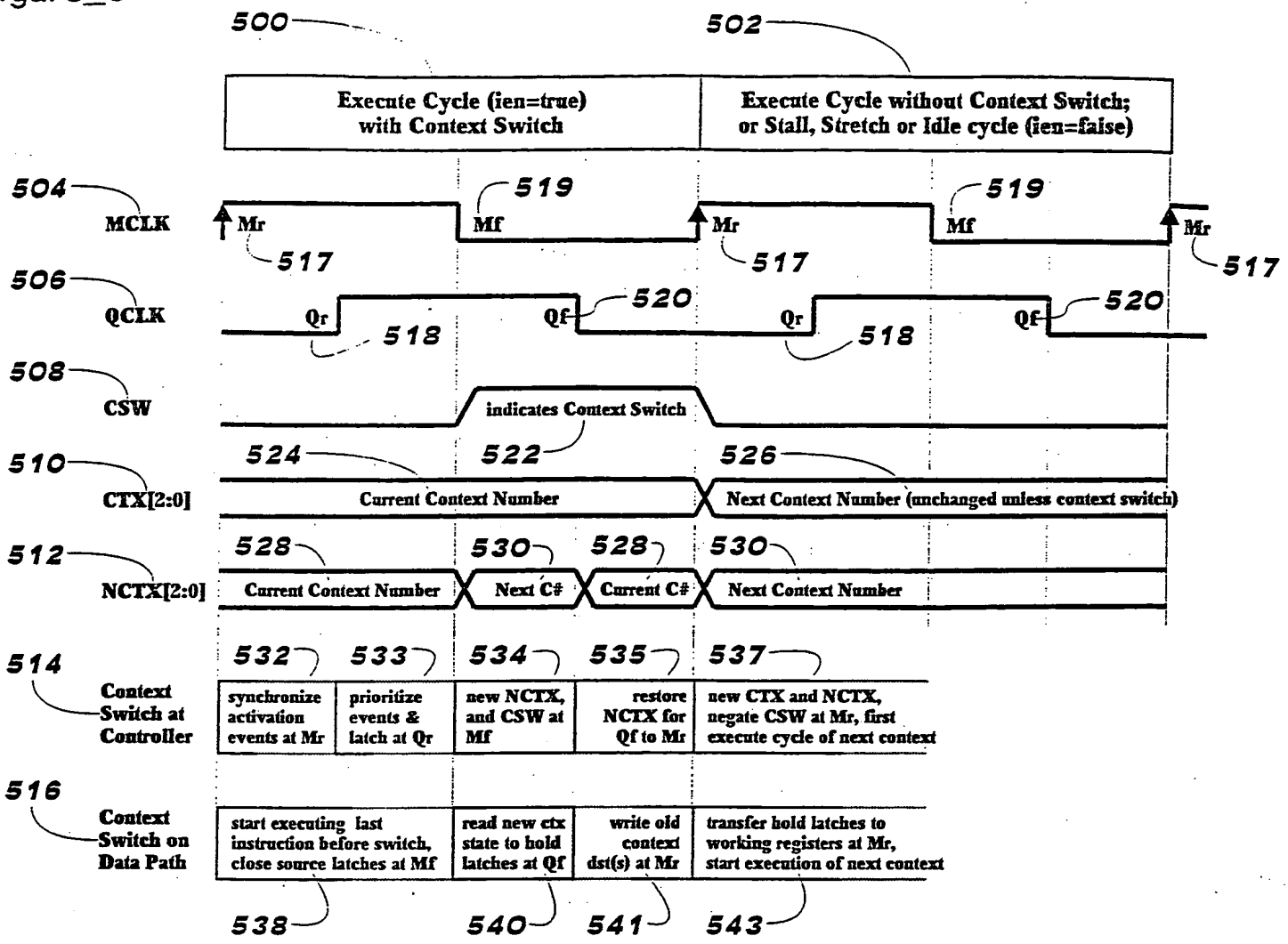

500
502
Execute Cycle (ien=true) with Context Switch
Execute Cycle without Context Switch; or Stall, Stretch or Idle cycle (ien=false)
504
MCLK
Mr
Mf
519
517
506
QCLK
Qr
Qf
520
518
508
CSW
indicates Context Switch
522
510
CTX[2:0]
524
Current Context Number
526
Next Context Number (unchanged unless context switch)
512
NCTX[2:0]
528
530
Current Context Number
Next C#
Current C#
Next Context Number
514
Context Switch at Controller
532
533
534
535
537
synchronize activation events at Mr
prioritize events & latch at Qr
new NCTX, and CSW at Mf
restore NCTX for Qf to Mr
new CTX and NCTX, negate CSW at Mr, first execute cycle of next context
516
Context Switch on Data Path
start executing last instruction before switch, close source latches at Mf
read new ctx state to hold latches at Qf
write old context dst(s) at Mr
transfer hold latches to working registers at Mr, start execution of next context
538
540
541
543

Figure_10

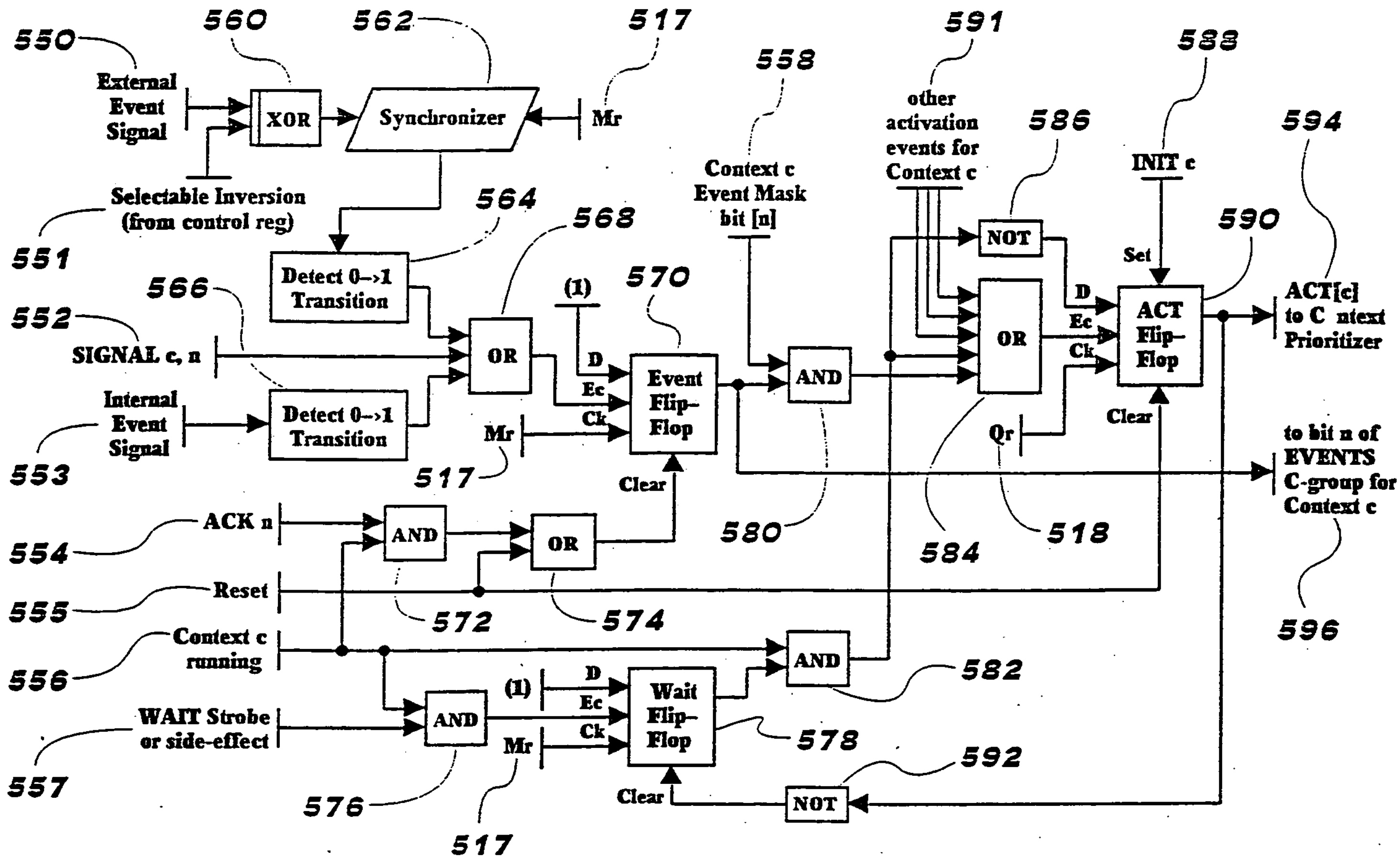

550
External Event Signal
560
XOR
562
Synchronizer
517
Mr
Selectable Inversion (from control reg)
551
564
Detect 0-->1 Transition
568
552
SIGNAL c, n
566
OR
(1)
570
D
Ec
Ck
Event Flip-Flop
Mr
Internal Event Signal
553
Detect 0-->1 Transition
517
Clear
558
Context c Event Mask bit [n]
AND
580
591
other activation events for Context c
586
NOT
OR
584
588
INIT c
Set
590
ACT Flip-Flop
Clear
Qr
518
594
ACT[c] to C ntext Prioritizer
to bit n of EVENTS C-group for Context c
596
554
ACK n
AND
OR
572
574
555
Reset
556
Context c running
AND
582
557
WAIT Strobe or side-effect
AND
576
(1)
Mr
517
Wait Flip-Flop
578
Clear
NOT
592

Figure_11

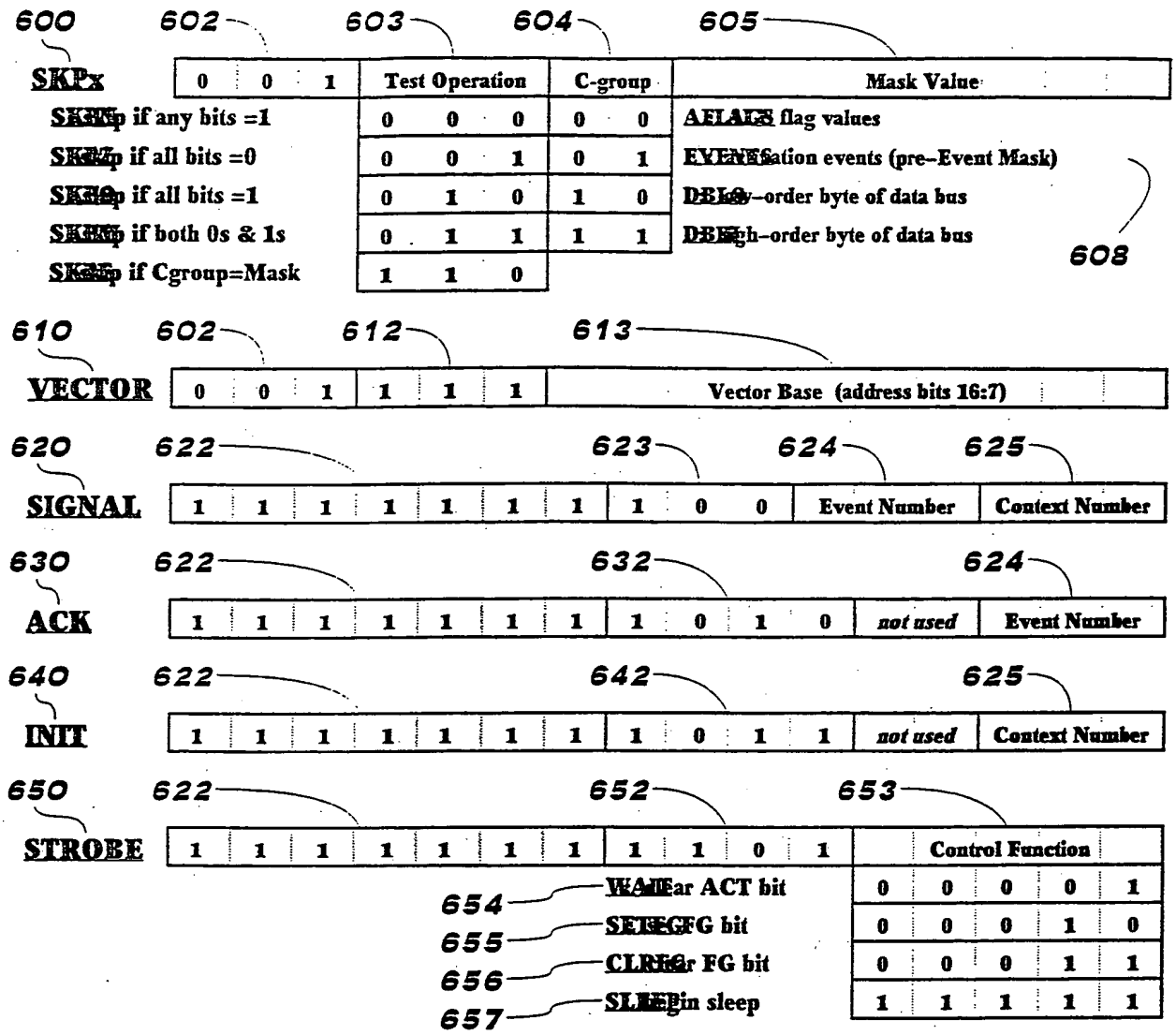

600
602
603
604
605
SKPx
0 0 1
Test Operation
C-group
Mask Value
Skip if any bits =1
0 0 0
0 0
AFLAGS flag values
Skip if all bits =0
0 0 1
0 1
EVENTS ation events (pre–Event Mask)
Skip if all bits =1
0 1 0
1 0
DBL0 –order byte of data bus
Skip if both 0s & 1s
0 1 1
1 1
DBL1 gh–order byte of data bus
Skip if Cgroup=Mask
1 1 0
608
610
602
612
613
VECTOR
0 0 1 1 1 1
Vector Base (address bits 16:7)
620
622
623
624
625
SIGNAL
1 1 1 1 1 1 1 1 0 0
Event Number
Context Number
630
622
632
624
ACK
1 1 1 1 1 1 1 1 0 1 0
not used
Event Number
640
622
642
625
INIT
1 1 1 1 1 1 1 1 0 1 1
not used
Context Number
650
622
652
653
STROBE
1 1 1 1 1 1 1 1 1 0 1
Control Function
654
Wa ear ACT bit
0 0 0 0 1
655
SE G FG bit
0 0 0 1 0
656
CLR r FG bit
0 0 0 1 1
657
SL in sleep
1 1 1 1 1

Figure_12

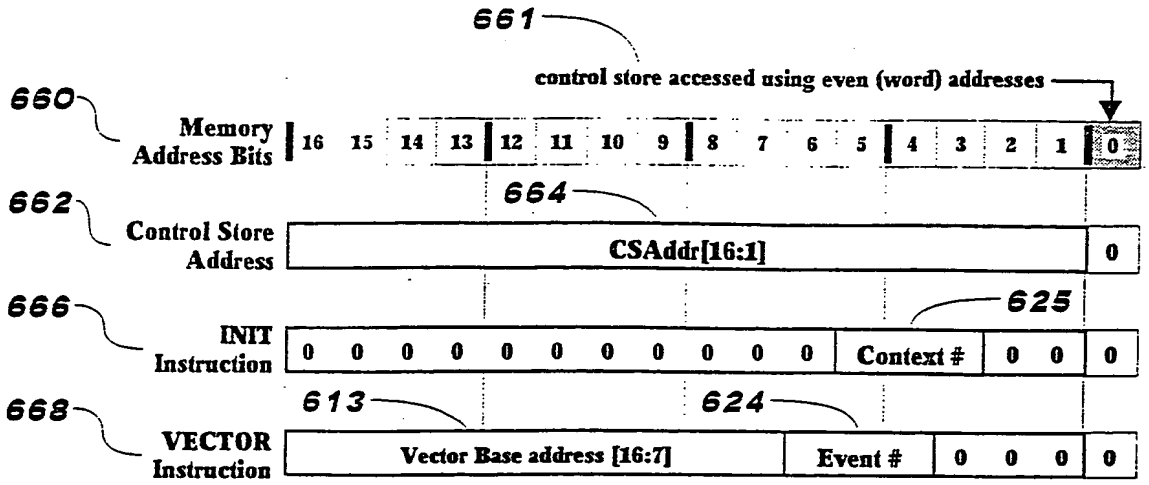

661
control store accessed using even (word) addresses
660
Memory Address Bits
16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
662
Control Store Address
664
CSAddr[16:1]
0
666
INIT Instruction
625
0 0 0 0 0 0 0 0 0 0 0
Context #
0 0 0
668
VECTOR Instruction
613
624
Vector Base address [16:7]
Event #
0 0 0 0

Figure_13

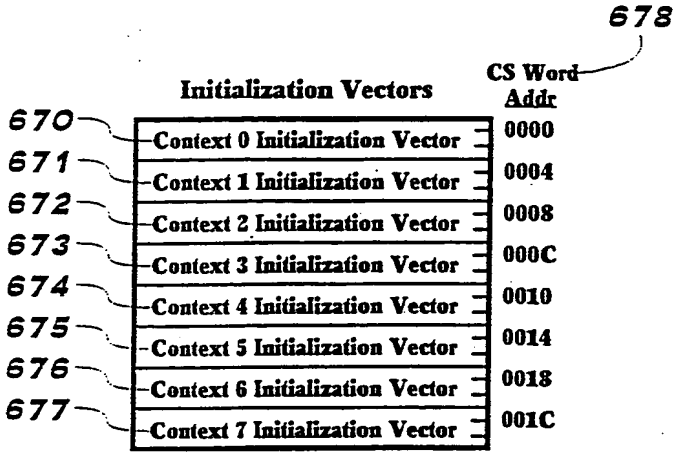
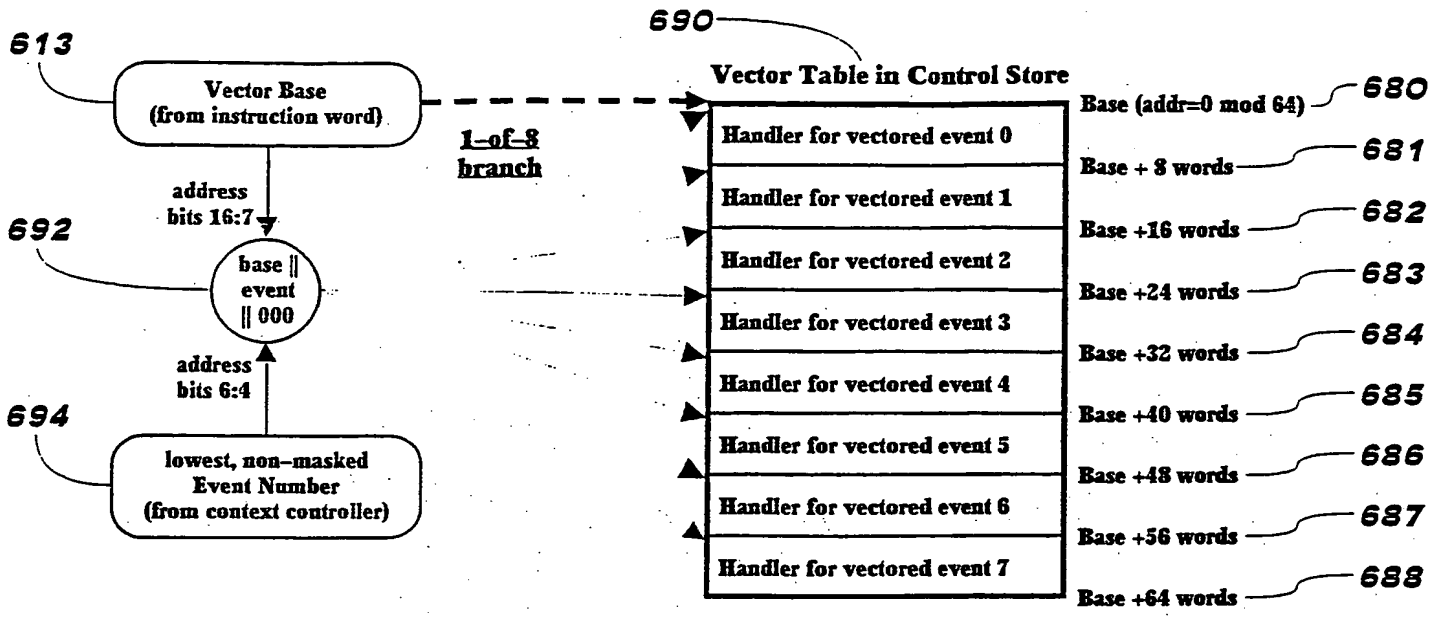

678
Initialization Vectors
CS Word Addr
670
Context 0 Initialization Vector
0000
671
Context 1 Initialization Vector
0004
672
Context 2 Initialization Vector
0008
673
Context 3 Initialization Vector
000C
674
Context 4 Initialization Vector
0010
675
Context 5 Initialization Vector
0014
676
Context 6 Initialization Vector
0018
677
Context 7 Initialization Vector
001C

Figure_14


613
Vector Base
(from instruction word)
address
bits 16:7
692
base ||
event
|| 000
address
bits 6:4
694
lowest, non-masked
Event Number
(from context controller)
1-of-8
branch
690
Vector Table in Control Store
Handler for vectored event 0
Handler for vectored event 1
Handler for vectored event 2
Handler for vectored event 3
Handler for vectored event 4
Handler for vectored event 5
Handler for vectored event 6
Handler for vectored event 7
Base (addr=0 mod 64)
680
Base + 8 words
681
Base +16 words
682
Base +24 words
683
Base +32 words
684
Base +40 words
685
Base +48 words
686
Base +56 words
687
Base +64 words
688